Platform-Independent Debugging of Physical Interaction and Signal Flow Models

Mehdi Dadfarnia[1] Raphael Barbau[2]

[1]National Institute of Standards & Technology

[2]Engisis, LLC

April 9, 2019

2019 IEEE International Systems Conference

Table of Contents

1 Intro

2 Background

3 Debugging Methods

4 Conclusions & Future Work

5 References & Backup Slides

Table of Contents

1. Intro
2. Background
3. Debugging Methods
4. Conclusions & Future Work
5. References & Backup Slides

Introduction

- Debugging procedures for integrated systems modeling and simulation tools
- Integration via SysPhS (publicly available standard)
  - Platform-independent integration of SysML with physical interaction and signal flow simulation tools
- Physical interaction & signal flow (PISF) models:
  - Interconnected components (system structure)
  - Energy/information exchanges between components (system behavior)

Systems Modeling & Simulation Tools

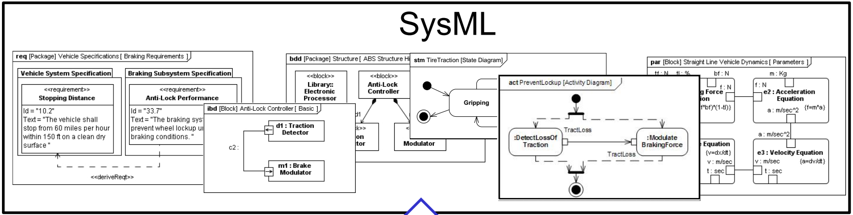
Systems modeling languages

- Organize & coordinate analysis focusing on subsets of components and interconnections
- Higher-level abstraction of component physical interactions & signal flows
- Example: SysML

Equation-based simulation languages

- Teams focus on separate subsets of components and interconnections with discipline-specific models
- Experiment with systems without physically building them
- Examples: Modelica, Simulink/Simscape

Integrating Systems Modeling & Simulation Tools

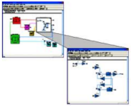

SysML
req [Package] Vehicle Specifications [Braking Requirements]
Vehicle System Specification
Braking Subsystem Specification
<<requirement>>
Stopping Distance
Id = "10.2"
Text = "The vehicle shall stop from 60 miles per hour within 150 ft on a clean dry surface"
<<requirement>>
Anti-Lock Performance
Id = "33.7"
Text = "The braking sys prevent wheel lockup u braking conditions."
<<deriveReqt>>
bdd [Package] Structure [ABS Structure Hi
<<block>>
Library:: Electronic Processor
<<block>>
Anti-Lock Controller
Modulator
ibd [Block] Anti-Lock Controller [Basic]
d1 : Traction Detector
c2 :
m1 : Brake Modulator
stm TireTraction [State Diagram]
Gripping
act PreventLockup [Activity Diagram]
:DetectLossOf Traction
TractLoss
:Modulate BrakingForce
TractLoss
par [Block] Straight Line Vehicle Dynamics [Parameters]
bf : N
m : Kg
f : N
g Force
e2 : Acceleration Equation
a : m/sec^2
{f=m*a}
a : m/sec^2
Equation
{v=dx/dt}
v : m/sec
t : sec
e3 : Velocity Equation
v : m/sec
{a=dv/dt}
t : sec

SysPhS

PI & SF simulation languages & tools

MATLAB SIMULINK

MODELICA OpenModelica



NATIONAL INSTRUMENTS LabVIEW

Scilab

Exchanging models of physical and informational behaviors

Goals

- Find errors in system models before they propagate to simulation models on multiple platforms
  - Does not require translating fixes to simulation models back into system models
  - Verifies system models before discipline-specific experts focus on their subsets
  - Debug in higher abstraction system models before translation to lower abstraction, domain-specific simulations
- Identify causes of failure to:
  - Execute simulation files translated from system models
  - Get expected results from simulation execution

Table of Contents

1. Intro
2. Background
3. Debugging Methods
4. Conclusions & Future Work
5. References & Backup Slides

Types of Errors & Debugging Techniques for PISF Models

Errors that cause simulation failures

- Inconsistent underlying system of equations & erroneous symbolic transformations
- Examples: overconstrained or underconstrained equations; equations that would divide by zero; functions called outside their domain

Errors that cause simulations to produce unintended results

- Desired system behavior different from simulation execution
- Examples: variable-values outside bounds; incorrect equations, function calls or initialization values; equation solver-specific behaviors

Types of Errors & Debugging Techniques for PISF Models

- Debugging procedure depends on type of error
- **Static Debugging:** Identify errors that cause failure to compile or simulate by tracing symbolic transformations
- **Dynamic Debugging:** Identify errors that cause simulations to produce unexpected results by comparing executions to static traces
- **Challenge:** Identifying errors in (bidirectional) physical interactions
  - No ordered execution of command sequences
  - More difficult in systems models: higher abstraction, fewer established debuggers

State-of-the-art Debugging Techniques for PISF Models

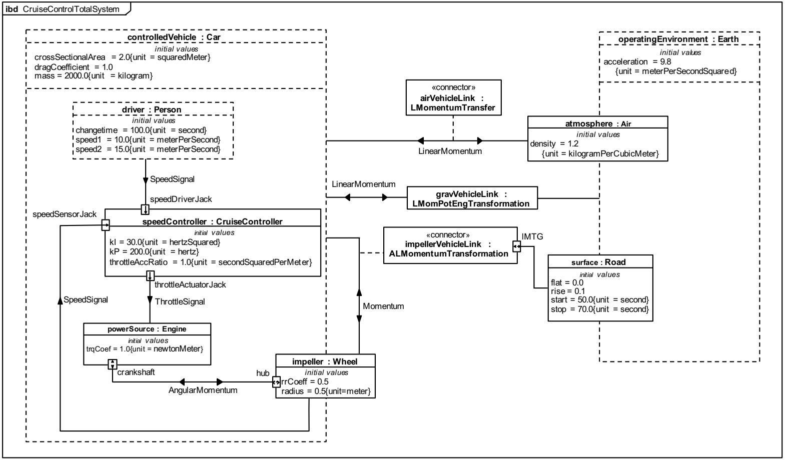
Debugging physical interactions

- Modelica community: integrate traditional debugging (breakpoints, stepping); determine balance of equation systems; isolate data flow slices
  - Integrate static debugging output into dynamic debugger with variable explorer at simulation run-time (Sjolund et al. 2014)

Debugging systems models

- Architecture for systems model debugger focusing on visualizing variables at simulation run-time rather than tracing symbolic transformations (Canedo and Shen, 2013)

Modeling Physical Interactions and Signal Flows with SysPhS



IBD for a vehicle cruise control system

Modeling Physical Interactions and Signal Flows with SysPhS

Key modeling elements:

- SysML internal block diagram, SysPhS extension
- Parts, ports, and components model system components
- Connectors model interactions for PI&SF
  - Mathematical relationships via parametric diagrams
  - Flow properties: specify the kind of signal or conserved physical substance (& their variables) flowing through boundary of component
  - *PhSConstant* & *PhSVariable* property stereotypes
  - Mathematical relationship between variables of flow properties: physical interactions vs signal flows; transformations across connectors
- *Implication for debugging: trace chains of connectors between components for the symbolic transformations*

Table of Contents

1 Intro

2 Background

3 Debugging Methods

4 Conclusions & Future Work

5 References & Backup Slides

Overview

1. **Preprocessing**: Makes debugging complex models more manageable
2. **Static Debugging:** Identifies errors in compiling or simulating the platform model
3. **Dynamic Debugging:** Identifies errors in simulation behavior
    - Prior static debugging recommended, not required

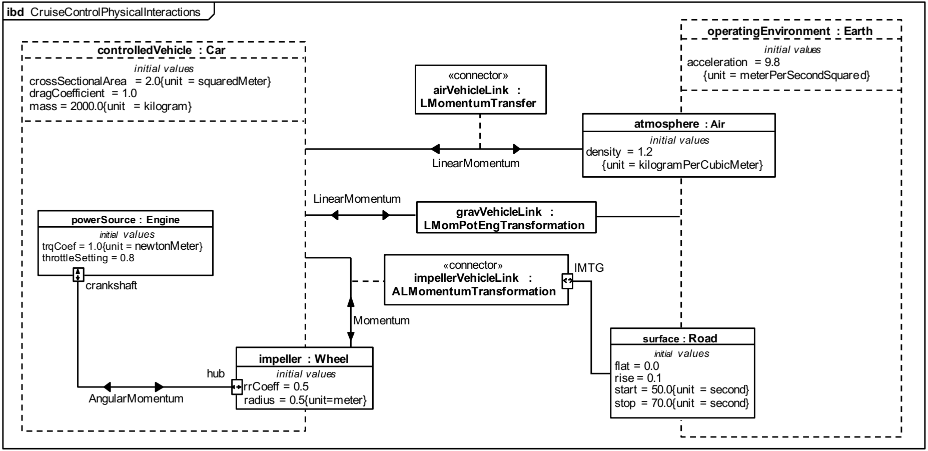
Preprocessing: Simplifying SysML Models for Debugging

- Decompose into smaller models; replicate fixes in complete model
  - Physical interactions-only & signal flows-only models

Steps to a physical interactions-only model:

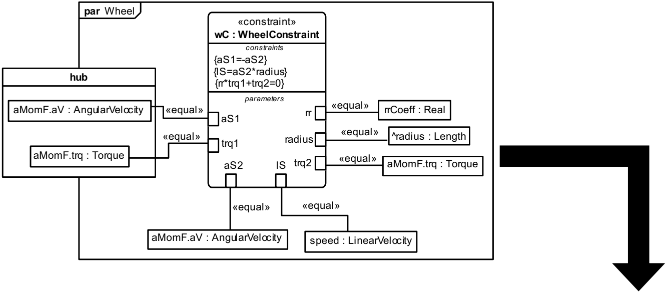
- IBD: Eliminate non-physical interaction connectors & their connected parts/ports
- Parametric diagrams:
  - Eliminate equations determining variables bound to signal *out*-flow properties
  - Eliminate part/port properties only bound to these equations
  - Replace equation variables bound to signal *in*-flow properties with constants
- Similar steps for signal flow model (see backup slides)

Preprocessing: Simplifying SysML Models for Debugging

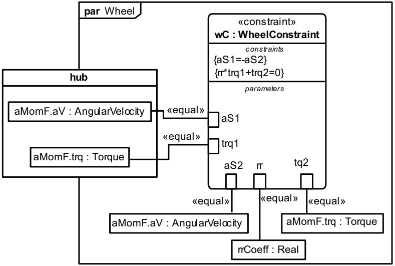


Only **physical interactions**

Preprocessing: Simplifying SysML Models for Debugging



Parametric diagram changes for **physical interactions** model



Static Debugging for Failure to Execute Simulation

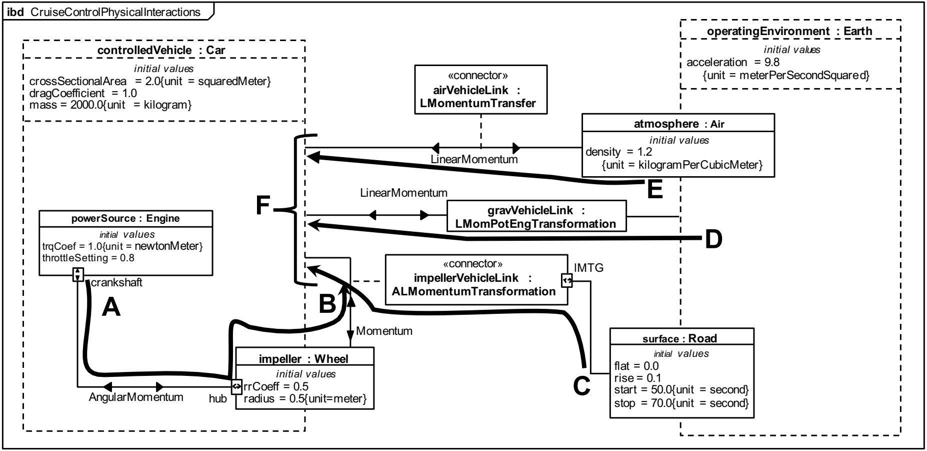
- Trace & bookkeep symbolic transformations:
  - Chains of connectors between components in IBDs
  - Constraint equations in parametric diagrams
- Known vs unknown variables
  - Variables known before simulation: bindings to constant properties, initial values, & time values
- Note: full trace must go through all connectors and parts/ports
- Replicate changes in original model

Static Debugging for Failure to Execute Simulation

Selecting a trace starting point:

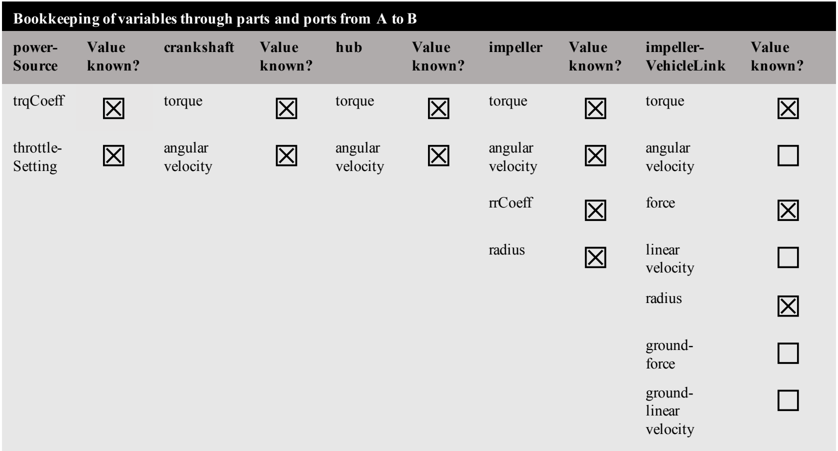
- Physical interaction-only models: Part that initiates rest of interactions (car engine)
  - If multiple candidate parts or ports; select arbitrarily
  - For non-obvious initiators: inspect parametric diagrams (more constants than variables)
- Similar steps for signal flow-only models (see backup slides)

Static Debugging for Failure to Execute Simulation



Traces from starting point A (initiating interaction part or port)

Static Debugging for Failure to Execute Simulation

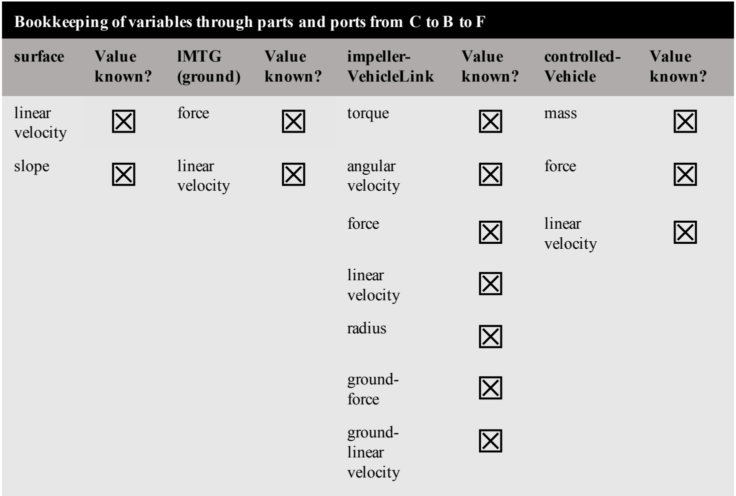


Bookkeeping of variables through parts and ports from A to B

power-Source	Value known?	crankshaft	Value known?	hub	Value known?	impeller	Value known?	impeller-VehicleLink	Value known?
trqCoeff	☒	torque	☒	torque	☒	torque	☒	torque	☒
throttle-Setting	☒	angular velocity	☒	angular velocity	☒	angular velocity	☒	angular velocity	☐
						rrCoeff	☒	force	☒
						radius	☒	linear velocity	☐
								radius	☒
								ground-force	☐
								ground-linear velocity	☐

Bookkeeping to track value assignments
-Added benefit: spotting erroneous equations

Static Debugging for Failure to Execute Simulation



Bookkeeping of variables through parts and ports from C to B to F

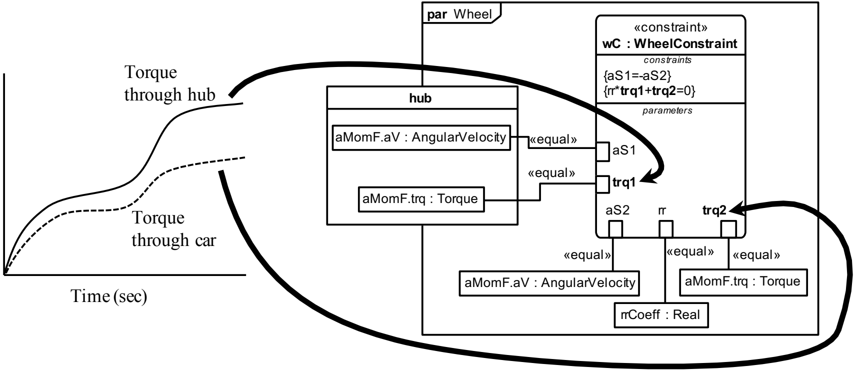
surface	Value known?	lMTG (ground)	Value known?	impeller-VehicleLink	Value known?	controlled-Vehicle	Value known?
linear velocity	☒	force	☒	torque	☒	mass	☒
slope	☒	linear velocity	☒	angular velocity	☒	force	☒
				force	☒	linear velocity	☒
				linear velocity	☒		
				radius	☒		
				ground-force	☒		
				ground-linear velocity	☒		

Bookkeeping and value assignment

Dynamic Debugging for Unexpected Simulation Results

- Simulation deviates from modeler expectations
- Bookkeep simulated values & compare to variables bound to flow properties
  - Understand flows of conserved substances and signals
  - Related via connectors & constraints
- Replicate changes in original systems model; test again
  - Multiple simulation platform testing for robustness
- Simplify further: temporarily remove components until simulation behaves as intended
  - Incrementally restore, simulate, & check model

Dynamic Debugging for Unexpected Simulation Results



Relationship between simulated variables & flow properties

Sufficiently long simulation run

Dynamic Debugging for Unexpected Simulation Results

Models with physical interactions

- Start tracing from interaction initiator points
- Simulated values correspond to flow property variables at connector endpoints
    - Flow rate & Potential to flow
- Compare simulated values to those related to the same part or connector
- Ensure simulated values reflect intended mathematical transformations
- Replace parts that have only *out*-flow properties with constants
- Similar steps for signal flow-only models (see backup slides)

Table of Contents

1. Intro
2. Background
3. Debugging Methods
4. Conclusions & Future Work
5. References & Backup Slides

Conclusions & Future Work

- Debugging systems models of bidirectional PI&SF
  - Complements existing debugging techniques on simulation platforms
- Identify causes of simulation/compilation failures and incorrect simulations
- Static debugging & dynamic debugging
- SysPhS enables fixes to more-abstract systems model before errors cascade to translated simulations
- Future considerations:
  - Debugging errors that only occur when physical interaction and signal flow are combined
  - User-friendly interface to visualize model translation
  - Provide explicit mapping between system components and simulation structures: names, locations, simulation history, etc

More Information

- An Improved Method of Physical Interaction and Signal Flow Modeling for Systems Engineering:
  - Paper: https://www.nist.gov/publications/improved-method-physical-interaction-and-signal-flow-modeling-systems-engineering
  - Slides: https://flumes.iei.liu.se/modprod/modprod2017_proceedings/modprod2017-keynote-ConradBock_SysML-PISF-PhysicalInteraction.pdf
- An Extension of the Systems Modeling Language for Physical Interaction and Signal Flow Simulation:
  - https://www.nist.gov/publications/extension-systems-modeling-language-physical-interaction-and-signal-flow-simulation
- Standard: https://www.omg.org/spec/SysPhS
- Implementation: https://github.com/usnistgov/saismo/releases/download/sysphs/sysphs1.0.zip

Thank you. Questions?

Table of Contents

1. Intro
2. Background
3. Debugging Methods
4. Conclusions & Future Work
5. References & Backup Slides

References

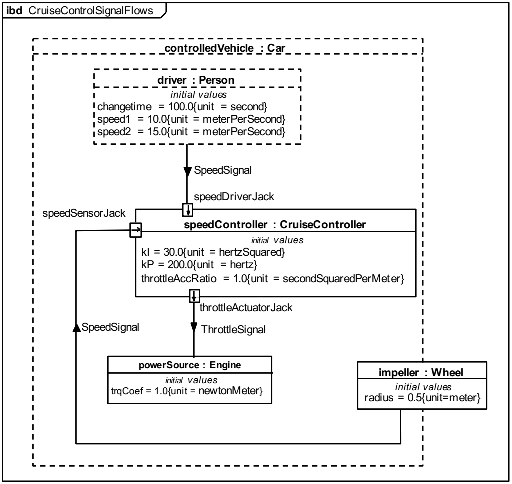
- M. Sjolund, F. Casella, A. Pop, A. Asghar, P. Fritzson, W. Braun, L. Ochel, and B. Bachmann, "Integrated Debugging of Equation-Based Models," in *10th international Modelica Conference*, Lund Sweden, March 2014, pp. 195-204.
- A. Candeo, and L. Shen, "Functional Debugging of Equation-Based Languages," in *5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Nottingham UK, April 2013, pp. 55-64.

Preprocessing a Signal Flows-Only Model

Steps:

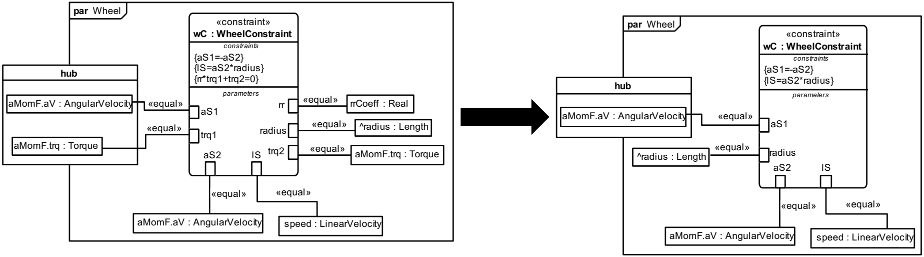
- IBD: Eliminate non-signal flow connectors & their connected parts/ports
- Parametric diagrams:
  - Eliminate equations that aren't used to determine variables bound to signal *out*-flow properties
  - Eliminate equations that have no bindings to signal *in*-flow properties
  - Eliminate part or port properties not bound to remaining equations
  - Replace any variables bound to physical interaction *inout*-flow properties

Preprocessing for Signal Flow-Only Models


ibd CruiseControlSignalFlows
controlledVehicle : Car
driver : Person
initial values
changetime = 100.0{unit = second}
speed1 = 10.0{unit = meterPerSecond}
speed2 = 15.0{unit = meterPerSecond}
SpeedSignal
speedDriverJack
speedSensorJack
speedController : CruiseController
initial values
kI = 30.0{unit = hertzSquared}
kP = 200.0{unit = hertz}
throttleAccRatio = 1.0{unit = secondSquaredPerMeter}
throttleActuatorJack
SpeedSignal
ThrottleSignal
powerSource : Engine
initial values
trqCoef = 1.0{unit = newtonMeter}
impeller : Wheel
initial values
radius = 0.5{unit=meter}

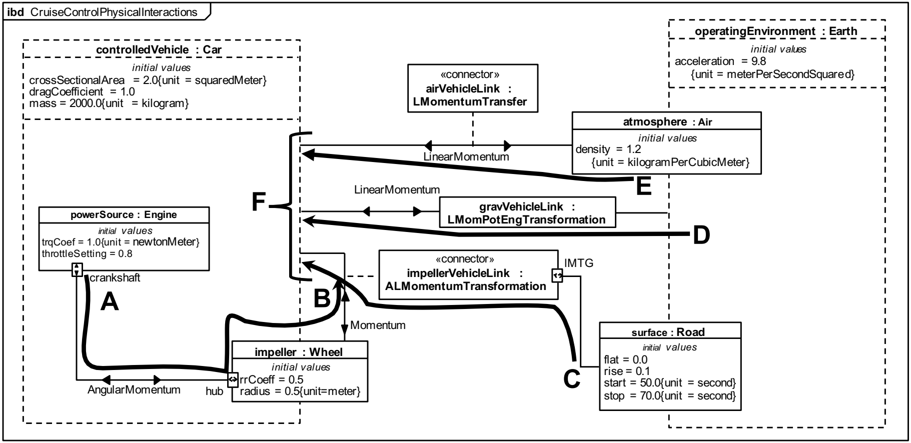
Only **signal flows** in cruise control model

Preprocessing for Signal Flow-Only Models



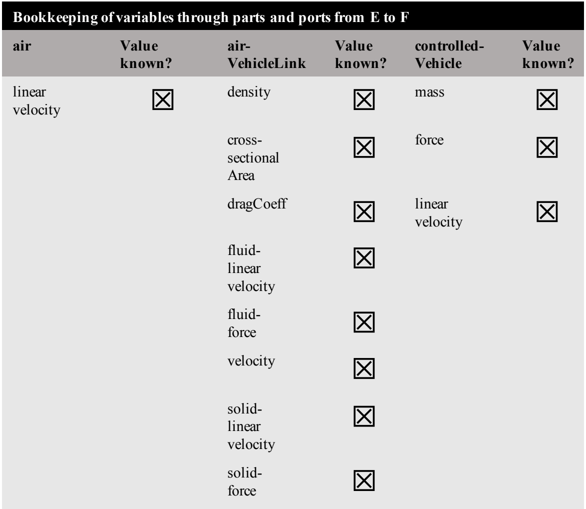
Parametric diagram changes for Wheel port

Static Debugging for Failure to Execute Simulation



Debugging through physical interactions, continued

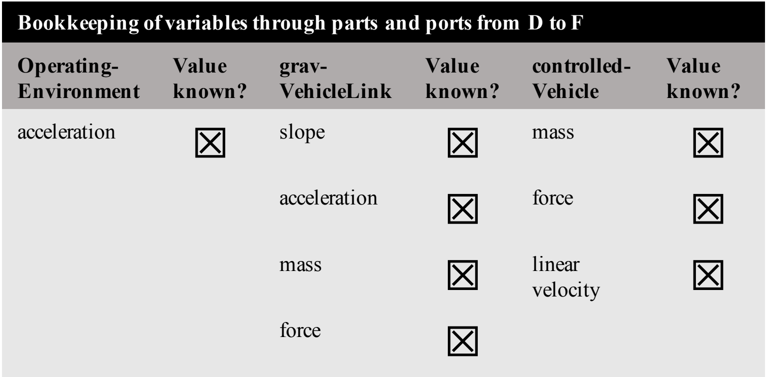
Static Debugging for Failure to Execute Simulation



Bookkeeping of variables through parts and ports from E to F

air	Value known?	air-VehicleLink	Value known?	controlled-Vehicle	Value known?
linear velocity	☒	density	☒	mass	☒
		cross-sectional Area	☒	force	☒
		dragCoeff	☒	linear velocity	☒
		fluid-linear velocity	☒		
		fluid-force	☒		
		velocity	☒		
		solid-linear velocity	☒		
		solid-force	☒		

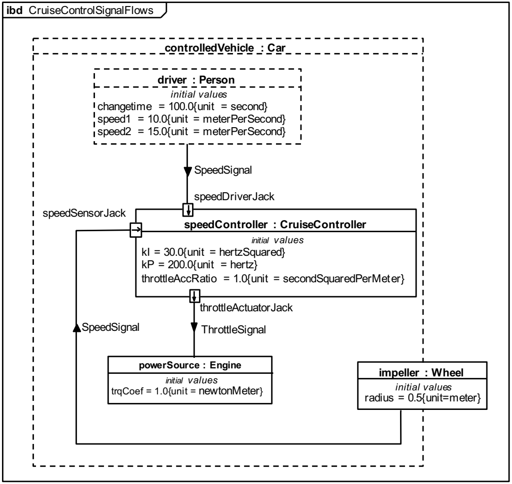
Static Debugging for Failure to Execute Simulation



Bookkeeping of variables through parts and ports from D to F

Operating-Environment	Value known?	grav-VehicleLink	Value known?	controlled-Vehicle	Value known?
acceleration	☒	slope	☒	mass	☒
		acceleration	☒	force	☒
		mass	☒	linear velocity	☒
		force	☒		

Static Debugging - Models with Signal Flows


ibd CruiseControlSignalFlows
controlledVehicle : Car
driver : Person
initial values
changetime = 100.0{unit = second}
speed1 = 10.0{unit = meterPerSecond}
speed2 = 15.0{unit = meterPerSecond}
SpeedSignal
speedDriverJack
speedSensorJack
speedController : CruiseController
initial values
kI = 30.0{unit = hertzSquared}
kP = 200.0{unit = hertz}
throttleAccRatio = 1.0{unit = secondSquaredPerMeter}
throttleActuatorJack
SpeedSignal
ThrottleSignal
powerSource : Engine
initial values
trqCoef = 1.0{unit = newtonMeter}
impeller : Wheel
initial values
radius = 0.5{unit=meter}

Similar to that of physical interactions;
Focus is on parts without an *in*-flow property

Dynamic Debugging - Models with Signal Flows

- Replace parts that have only *out*-flow properties with constants
- Signal flow-only models:
  - Similar to that of physical interactions, but start from parts without an *in*-flow property